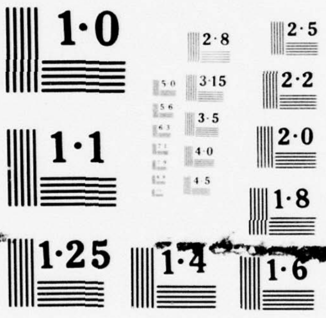

1·0
2·8
2·5
5·0
3·15
2·2
5·6
3·5
6·3
2·0
1·1
4·0
4·5
1·8
1·25
1·4
1·6

C

B.S.

COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

6

A DISCUSSION OF HARDWARE IMPLEMENTATION AND FABRICATION FOR AN AUTOMATIC TARGET CUEING SYSTEM.

12 46p.

11 31 Apr~~il 30,~~ 1977

This is the fourth quarterly status report on a program for Recognition Technology for a Smart Sensor, conducted by Westinghouse for U. Md.under Contract 15 DAAG 53-76-C-0138 with the U. S. Army Electronics Command, ~~Night~~ Vision Laboratory, Ft. Belvoir, Va. 22060

9 Quarterly rept. no. 4.

Prepared for

Computer Science Center
University of Maryland
College Park, Maryland 20742

By

Westinghouse Defense and Electronic Systems Center
Systems Development Division
Baltimore, Maryland 21203

403018

mt

TABLE OF CONTENTS

INTRODUCTION

This is the fourth quarterly status report on a program for RECOGNITION TECHNOLOGY FOR A SMART SENSOR, being conducted by Westinghouse Systems Development Division for the Computer Science Center, University of Maryland. The program consists of three phases, as follows:

Phase I	Task and Technology Review (3 months)
Phase II	Algorithm Selection and Test (9 months)
Phase III	Hardware Development (9 months)

This report covers the final third ~~3 months~~ of the Phase II effort. The report was prepared by Mr. Thomas Willett and Dr. Nathan Bluzer of Westinghouse. The Westinghouse program manager is Dr. Glenn E. Tisdale.

During the quarter, six meetings were held between members of the Maryland and Westinghouse teams. Mr. John Dehne and Dr. George Jones of NVL attended several of the meetings.

Westinghouse is concentrating on the hardware implementation and fabrication of the U. Md. algorithms for the focal plane and treating them as a system. This quarter continues the emphasis on implementation and fabrication. This quarter also marks the first time that more intelligent operators (image segmentation, feature extraction, and classification) have been embodied in CCD technology.

1.0 SYSTEM FLOW

This section describes a preferred set of algorithms developed by Maryland which tentatively comprises the first portion of a cueing system. A system flow chart is shown in Figure 1-1. A description of data flow and storage requirements is included.

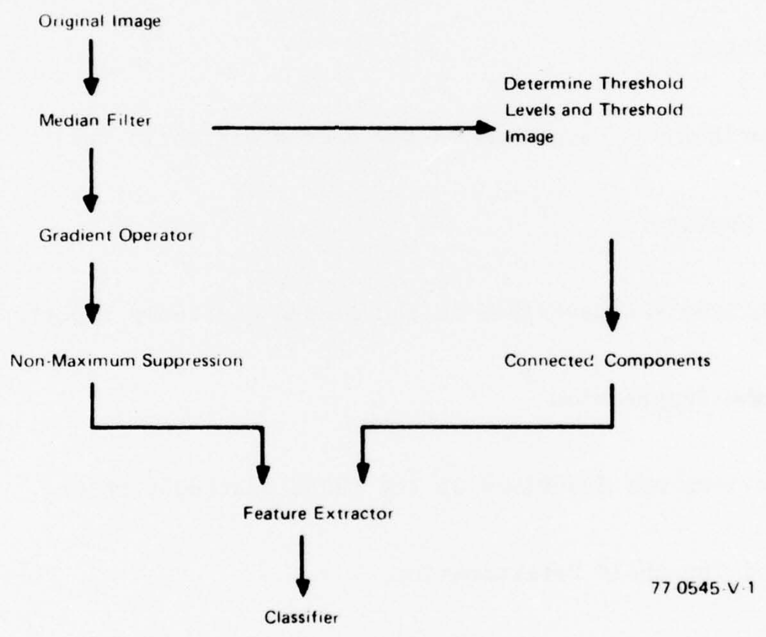


Figure 1-1. System Flow Chart

In general, the Median Filter acts to suppress noise. The Gradient Operator extracts edges which are then thinned by the Non-Maximum Suppression Algorithm. At the same time a set of gray levels is determined and the filtered image is thresholded at each gray level. A Connected Components Algorithm partitions the thresholded image into object regions. A matching Algorithm correlates perimeter points formed independently by the Non-Maximum Suppression and Connected Components Algorithms and a score is obtained. This score and several other algorithms form a set of Classification Logic.

1.1 Algorithms

A short description of each algorithm follows.

1.1.1 Median Filter

This algorithm was described in the second quarterly report.

1.1.2 Gradient Operator

This algorithm was described in the second quarterly report.

1.1.3 Non-Maximum Suppression

This algorithm was described in the third quarterly report.

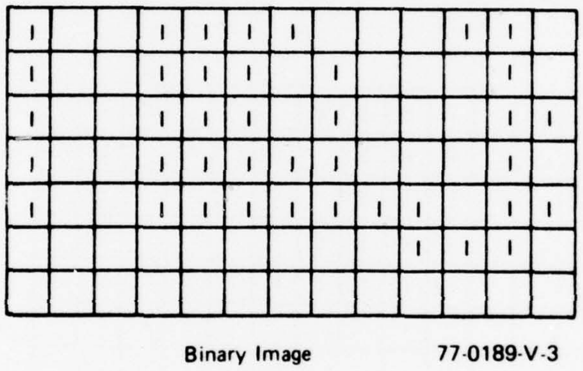
1.1.4 Gray Level Threshold Determination

U. Md. is evaluating several approaches to this determination; hardware implementation will be held in abeyance until completion.

1.1.5 Connected Components

This algorithm was described in the third quarterly report. Because the algorithm is the subject of the Hardware Implementation Section of this quarterly report, we repeat that description here. The purpose of the algorithm is to segment the image data stream into smaller domains. Each small domain includes a single object in the image plane. This algorithm distinguishes between objects and isolates regions so that statistics for Classification Logic can be obtained.

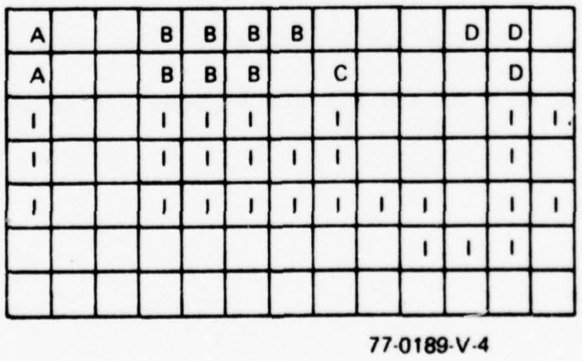
Assume that the original image has been thresholded and the result is in binary form with gray levels exceeding g_1 shown as 1's in Figure 1-2.



1			1	1	1	1				1	1	
1			1	1	1		1				1	
1			1	1	1		1				1	1
1			1	1	1	1	1				1	
1			1	1	1	1	1	1	1		1	1
									1	1	1	

Binary Image
77-0189-V-3

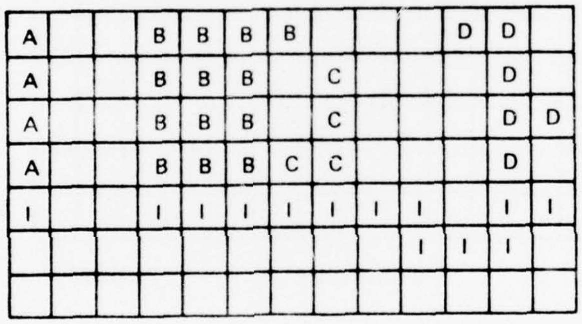
Figure 1-2a. Binary Image



A			B	B	B	B				D	D	
A			B	B	B		C				D	
1			1	1	1		1				1	1
1			1	1	1	1	1				1	
1			1	1	1	1	1	1	1		1	1
									1	1	1	

Figure 1-2b. Computations for Second Row

Two image lines are retained in memory so that each pixel can examine its neighbors to the left and right and above and below. No diagonal connections are permitted under this convention, and an adjacent (horizontal or vertical) pixel must be occupied in order to make a connection. No skips or gaps are allowed, and the computations start one pixel in from the edge. In Figure 1-2b, there are four distinct regions, A, B, C, and D. The only possible connection between regions B and C is through a diagonal, which is not allowed. Computations for the fourth row are seen in Figure 1-2c.



A			B	B	B	B				D	D	
A			B	B	B		C				D	
A			B	B	B		C				D	D
A			B	B	B	C	C				D	
I			I	I	I	I	I	I	I		I	I
									I	I	I	

Figure 1-2c. Computations for Fourth Row

Here, there is a connection between regions B and C and an equivalence statement, B = C, is carried along. At the end of the sixth row, there is another connection between C and D (C = D) and all the regions are completed as seen in Figure 1-2d.

A			B	B	B	B				B	B	
A			B	B	B		B				B	B
A			B	B	B		B				B	
A			B	B	B		B				B	
A			B	B	B	B	B	B	B		B	B
									B	B	B	

77-0189-V-6

Figure 1-2d. Completed Image

The areas of A, B, C and D are computed by cumulating the number of pixels assigned to each. The perimeter is calculated by cumulating the number of pixels assigned to each region which are neighbors of zeros, i.e., the neighbors did not exceed the gray level threshold, g_i.

1.1.6 Perimeter Match

This algorithm was described in the third quarterly report.

1.1.7 Feature Extraction

Maryland has begun to designate features to be extracted from the Connected Components Algorithm for classification. A tentative list of features, including perimeter match, contains specified average gray level over the target, perimeter extent, area, and maximum height and width.

1.2 Data Flow

We described the focal plane data flow in the third quarterly report and noted that the image plane was divided into 20 separate columns. Since the image was assumed to be 640 pixels wide, each column was slightly wider than 34 pixels to accommodate the edge point along each column. Specifically, assume we have a primitive operator which requires a 5 x 5 kernal and the result is placed in the center pixel of the 25 pixel array. This means that the operator in the first column can only process pixels 3-32. Also, the operator in the second column processes pixels 37-66 and there is a 4 pixel gap at the lines separating columns. One way to avoid this problem is to make the serpentine delay line slightly wider, i.e., send pixels 35 and 36 to column one operator so that it can process pixels 33 and 34. Further send pixels 33 and 34 to the column two operator so that it can process pixels 35 and 36. In this many, the vertical gaps in the image will be removed.

1.3 Storage Requirements

The inclusion of Feature Extraction Algorithms has increased the need for additional delay lines. This occurs in this way. In the Perimeter Match and Average Gray Level Algorithms, the image of edges and gray levels must be retained until the target has been isolated by the Connected Components Algorithm. Then the two are clocked pixel by pixel into the appropriate logic to obtain sums and comparisons. To the extent of the delay between Connected Components and the edges, gray levels will correspond the length of the delay line. This appears to be rather small.

2.0 HARDWARE IMPLEMENTATION

In the prior section we discussed system flow, algorithms, data flow and storage requirements for the Maryland design. In this section we shall discuss specific hardware techniques to perform the algorithms.

2.1 Connected Components

This algorithm distinguishes between objects and isolates them so that classification statistics can be gathered. In the description of the algorithm in Section 1.1.5, it is noted that equivalence statements were used to indicate that several branches of the same object had different labels or "colors". Let us repeat this idea with the objects shown in Figure 2-1; these objects are separate and must be recognized as such by the Algorithm. Visually, this task is readily performed since we perceive the entire image, i.e., we see the global picture. On the IR focal plane, these shapes are read out in series or parallel line-by-line and only local information about each shape is available at any particular time. Without any record keeping, the local information is insufficient to separate different objects. This assertion

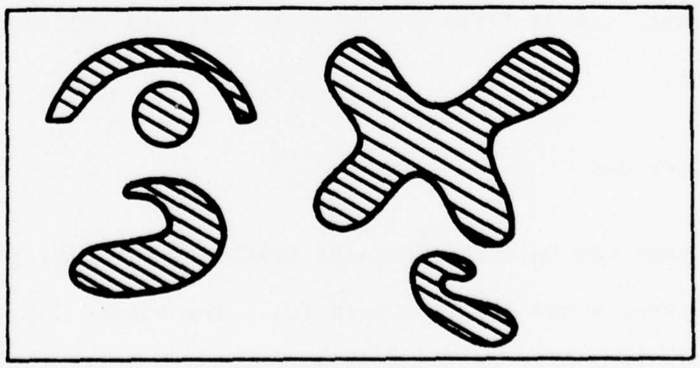


Figure 2-1. Five Objects 77-0545-V-2

can be illustrated if we attempt to label each shape by a different color as the IR array is read out. Assume the array is read out serially, one horizontal line at a time. If each number represents a different color, the result is shown in Figure 2-2.

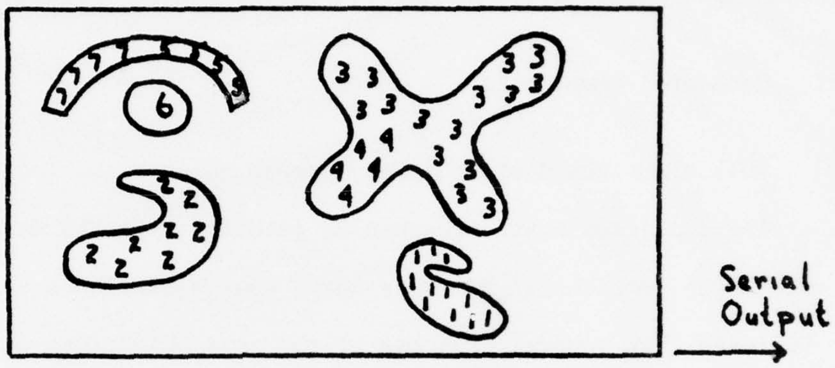


Figure 2-2. After Coloring

The resulting coloring does distinguish one shape from another, however it does not recognize differently colored portions of a single object as belonging to a single shape. If proper segmentation of the image is to occur, the shape segmentation must be improved to alleviate the "coloring operator's" shortcoming. Before pursuing this point, let us first describe the hardware implementation of the coloring operator.

2.1.1 Coloring Operator

We assume that the Coloring Operator processes a binary image, i.e., each pixel contains either a one (1) or a zero (0). The binary data stream will enter the Coloring Operator and emerge transformed into different colors or signal levels for different shapes. Read out of the binary picture will progress one horizontal line at a time starting with the bottom line and progressing upwards. Each horizontal line will be read out from right to left. Since the image data is read out serially, the Coloring Operator is a local operator. The coloring of each non zero element in the image plane will be done as shown in Figure 2-3. The Coloring Operator is a transformation from a binary picture to a colored one by a mapping T

$$T\ (A,\ B,\ M):\ C \rightarrow C^{1}$$

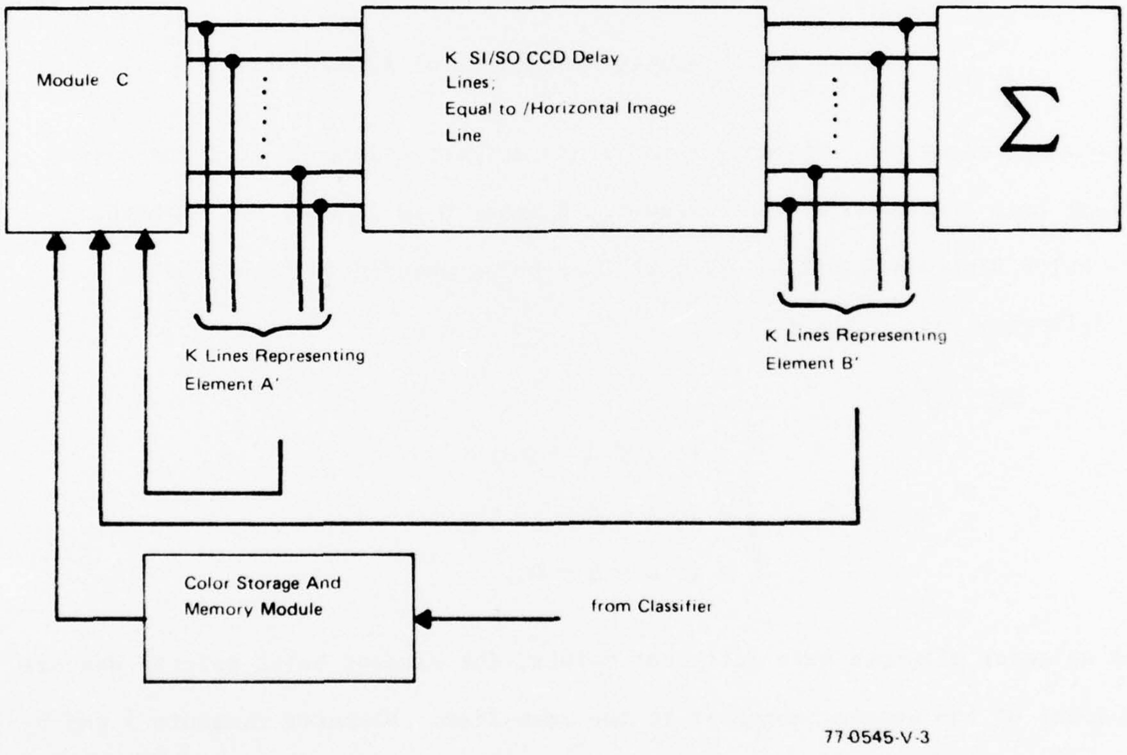

Module C
K SI/SO CCD Delay Lines;
Equal to /Horizontal Image Line
Σ
K Lines Representing Element A'
K Lines Representing Element B'
Color Storage And Memory Module
from Classifier
77-0545-V-3

Figure 2-3. Coloring Operator

where C^1 is the color of the transformed pixel C, the variables A and B represent nearest neighbors of C and M represents the color available from the Color Storage and Memory Module. The relative location of pixels A, B, and C in the image plane is shown in Figure 2-4. We define the coloring window as always containing

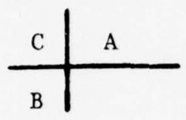


Figure 2-4. Relative Locations of A, B, C

these three elements. Elements A and B are nearest neighbors of C and have already been processed by the operator. Element B is located one horizontal line below elements C and A. Element C is being painted by CO according to the following rule (Module C):

For $C \neq 0$

$$C^1 = \begin{cases} A \text{ if } A \neq 0,\ B \neq 0 \\ B \text{ if } A = 0 \\ M \text{ if } A = B = 0. \end{cases}$$

when adjacent elements have different colors, the element being painted assumes the color of the nearest neighbor in the same line. Whenever elements A and B are zero and element C is not zero, element C^1 is given a new color from the CSMM. The CSMM will be a color bank providing colors not previously used in the neighborhood of the object. Painting of the binary picture with different colors will be accomplished by using a different number of charge quanta to represent various colors. The number of colors required is a function of target size, shape, and frequency of occurrence within the image plane. For purposes of the immediate discussion, we shall assume K colors are required, therefore each non zero pixel can require up to K memory locations to represent its color by a thermometer code (see Quantizer Section of the 3rd Quarterly).

The configuration of Module C is shown in Figure 2-5; note that there are K outputs C_K^1 for a single input C. Such a construction is required to reduce cummulative errors if analog implementation of Module C is attempted. Moreover, the configuration shown in Figure 2-5 is readily amenable to LSI and CCD technology. Each bit of C^1 output is governed by the expression,

For $C_K^1 \neq 0$

$$C_K^1 = \begin{cases} A_K^1 \text{ if } A_K \neq 0,\ B_K + 0 \\ B_K^1 \text{ if } A_K = 0 \\ M_K^1 \text{ if } A_K = B_K = 0 \end{cases}$$

The required input for operating Module C are C, A^1, B^1 and M^1. Input A^1 is readily obtained by tapping the first vector element stored in the SI/SO CCD delay line. Similarly, vector element B^1 is also obtained by tapping the output of the SI/SO CCD delay line. The construction of this delay line is readily feasible and has been addressed in the first quarterly report. The summing module Σ simply sums the charge present in all the K locations of vector B^1 thereby obtaining a quantized analog representation for different colors. This will prove of value later when we recolor the image. The only module not described is the Color Storage and Memory Module.

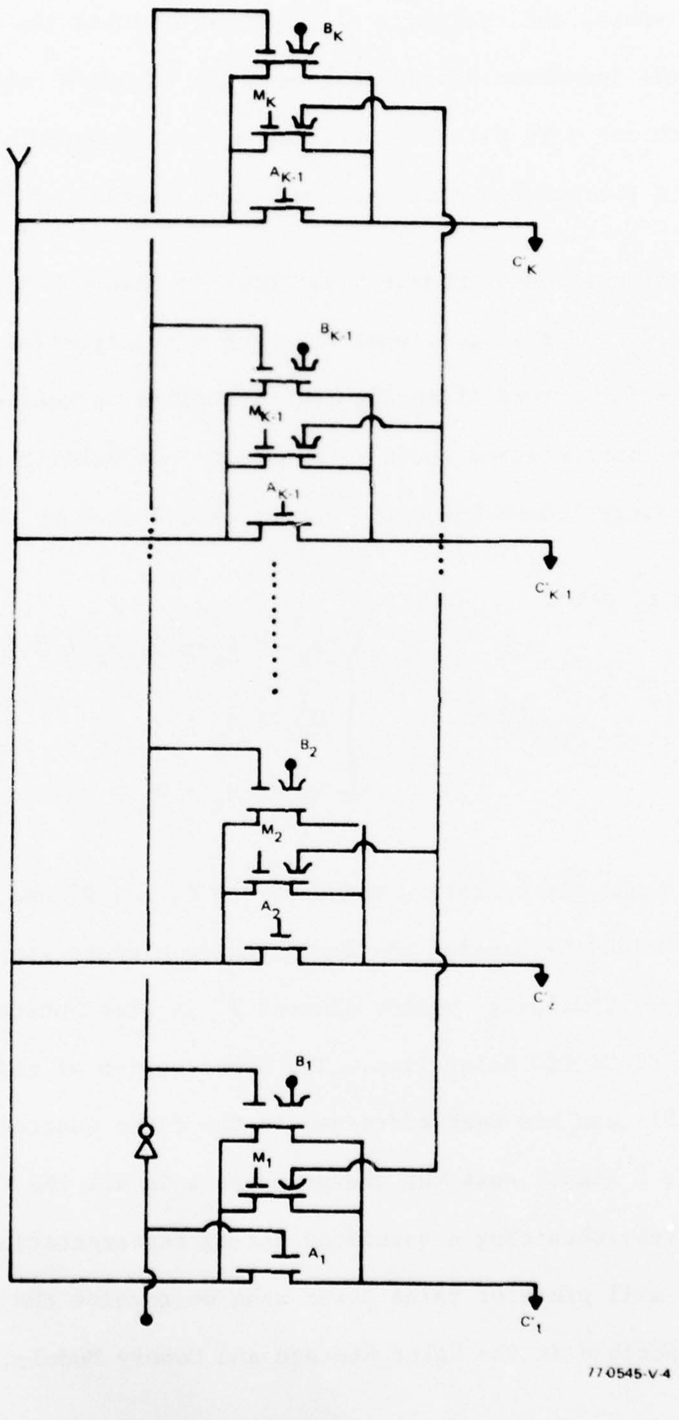


Figure 2-5. Details of Module C

This module will perform the tasks of keeping a record of the colors used during the process of painting a given frame, provide available colors for Module C to paint new shapes entering the Coloring Operator, and update the available color bank as colors become available again. Recall the description of the Quantizer and Sorter components of the Median Filter in the third quarterly report, for it appears that these components will comprise the Color Storage and Memory Module. The Quantizer takes an analog signal and produces a representation called a thermometer code; each color will have its own distinct thermometer code. A collection of unique colors are entered into a Sorter and clock pulses will cause the colors to exit the Sorter in a monotonic decreasing order; each clock pulse will cause a different color to exit. As long as a color is not re-entered into the Sorter, it is not available for further use. Once a color is re-entered into the sorter it becomes available for further use and its time of availability depends only on the relative magnitude of the colors within the Sorter. Then if a color is removed from the Sorter, applied to a target and not reused, i.e., not re-entered in the Sorter, until the target is classified, there is no danger of using the same color twice in a local area. Once a target is classified, the color is available again and is re-entered into the Sorter. This is the same as entering another pixel into the Median Filter.

It should be added that the Coloring Operator window used in this discussion was, in part, selected to aid the explanation. The shape of the window and logic which govern the operation may be modified with still the same end result but bowing to hardware constraints. For example, the arrangement described implies a single Coloring Operator which must process data at a rate of 1 megapixel/sec.; this may be possible since only Logic operations are involved. In any event, the Hardware Fabrication Analysis will provide more information.

Two problems remain in the implementation of the Connected Components Algorithm; the first is sorting the objects by color and sending each to a separate classifier, and the second is recoloring, with a single color, those objects which are composed of two or more colors (See Figure 2-2). All the modules described previously are now contained in the Coloring Operator shown in Figure 2-6 which is a block diagram of the entire Connected Components Algorithm.

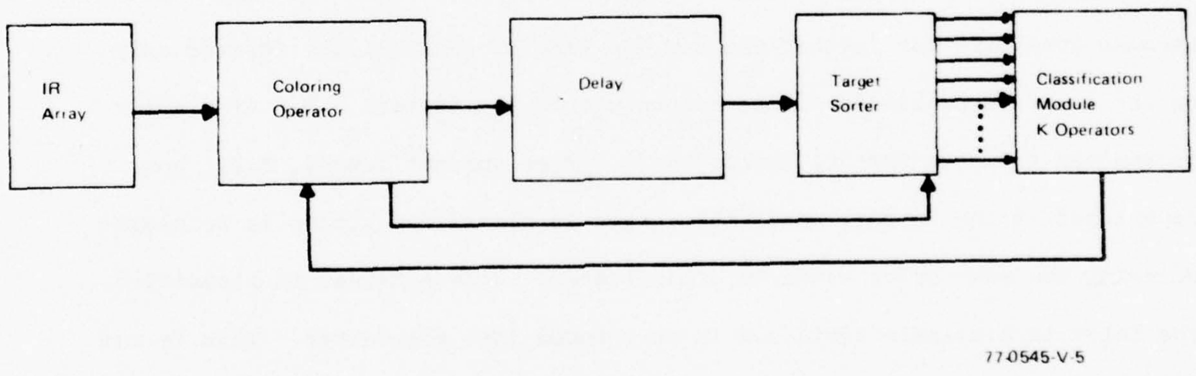


Figure 2-6. Connected Components Block Diagram

Suppose the Coloring Operator has colored an object as shown in Figure 2-7; we again assume that the Operator is working upward from the bottom of the image and one horizontal line at a time, from right to left. At the top

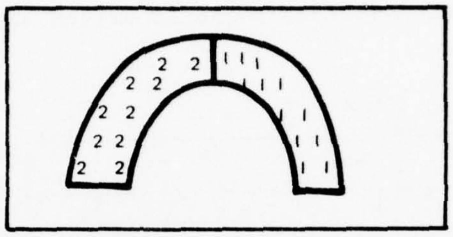


Figure 2-7. Multi-colored Object

of the arch, we would realize that the two colors are really part of the same object; this is equivalent to saying that the delay in Figure 2-6 must be as large as the largest expected target so as to encompass the target. This allows all adjacent colors within an object to be discovered before that object is lost from memory. We shall assume, for the sake of discussion, that the delay is 100 x 100 pixels. If one considers stand off ranges for helicopters and target sizes at those ranges, this is probably large. Anyhow, at the top of the arch in Figure 2-7, the Coloring Operator has found horizontal connectivity between pixels of different colors and now flags the Target Sorter as to the existence of a multi-colored object. We also make the assumption that the colors are in the form of analog signals which are being shifted from CCD site to CCD site through the delay. The Target Sorter detects the the color in a pixel location and begins sending the following data stream of the same color into a Classification Module corresponding to that color. Each Classification Module is reserved for one of the K colors. Having completed a general discussion of the implementation, let us proceed into some of the details.

2.2 Target Sorter

The Target Sorter Module is a matrix switch shown in Figure 2-8.

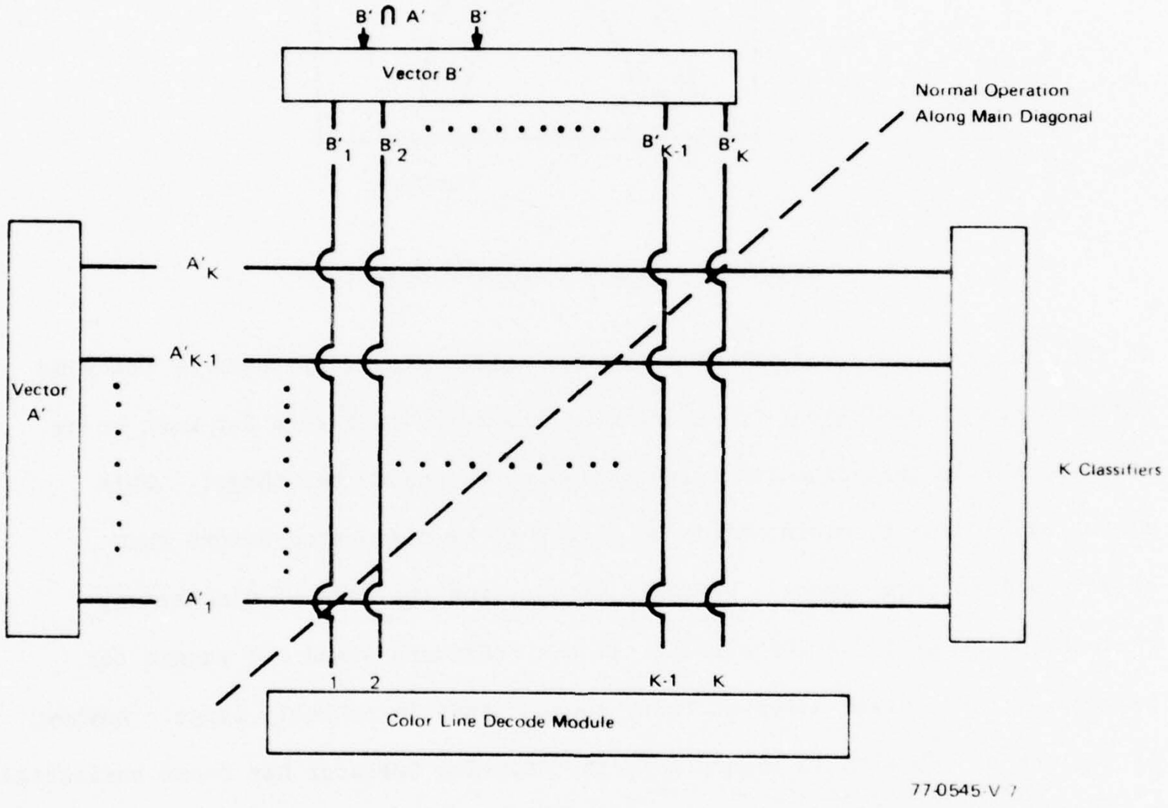


Figure 2-8. Target Sorter

The analog signal B^1, from the Coloring Operator, is quantized into K levels and the highest non-zero level bit i ($1 \leq i \leq K$) is identified; this i^{th} colored target is then sent to the i^{th} Classifier. So the Target Sorter via the matrix switch performs a number of specific tasks.

Each color data element, analog signal occupying one CCD site, is quantized into a K element vector. Only the highest subscript vector element is retained for each color. The highest element retained enables one vertical line of the K lines connected to the Color Line Decode Module. Each vertical line from the CCDM is connected via a switch to only one horizontal line going to the K classifiers. Normally the K^{th} CCDM line is connected to the K^{th} horizontal classifier line.

Quantizing the analog color signal requires a Quantizer described in the third quarterly report in regard to the Median Filter. Enabling the particular vertical line to the CCDM is not difficult since the largest non zero element is followed by a zero element unless it is the K^{th} element, in which the K-1 and K^{th} elements are both ones. Figure 2-9 shows an enabling circuit for the vertical CCDM lines.

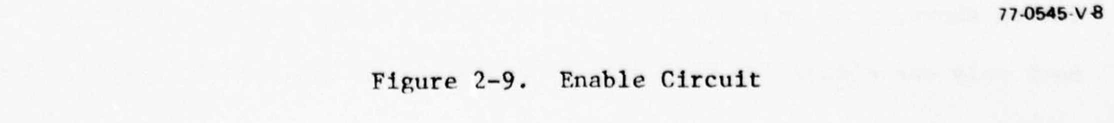


Figure 2-9. Enable Circuit

The color element corresponding to the vertical CCDM line is fed directly to the AND gate. The complement of the next higher element is also fed to the AND gate; this gives the non-zero, zero combination. For the K^{th} line, the K-1 element is fed uncomplemented to the K^{th} AND gate. This is the normal mode of operation for the Target Sorter; next we consider the case when portions of the same target have different colors.

We will show how the outputs from vector A^1 and B^1 modules can modify the diagonal interconnection between the CCDM and the K classifiers. A switch is located at each intersection between the horizontal and vertical lines in Figure 2-8. The function of this switch is to repaint each target into a single color and this switch will be reset when the classifier has finished a given color. For example, if vector A^1 has color i, $1 \leq i \leq K$, and vector B^1 has color j, $i \leq j \leq K$, $i \neq j$, only the i, j latch switch is enabled. Once this happens, all the data elements painted with the j^{th} color will be merged with the data elements painted with the i^{th} color. The i and j elements will be channeled to a single classifier. This color merge operation is achieved by connecting the i and j outputs from the CCDM unit via switches to the i^{th} Classifier.

To a large extent, then, merging different colors within the same target can be achieved by selecting appropriate switches and connecting them at the intersection between the vertical and horizontal lines of Figure 2-8. A diagram of such a switch is shown in Figure 2-10. Two types of switches are required for the Target Sorter; switches on the main diagonal will be different from those placed off the main diagonal. From the switch configuration shown, only one switch will be on in any given column. Therefore, at most only one classifier line will be connected to each vertical line. The

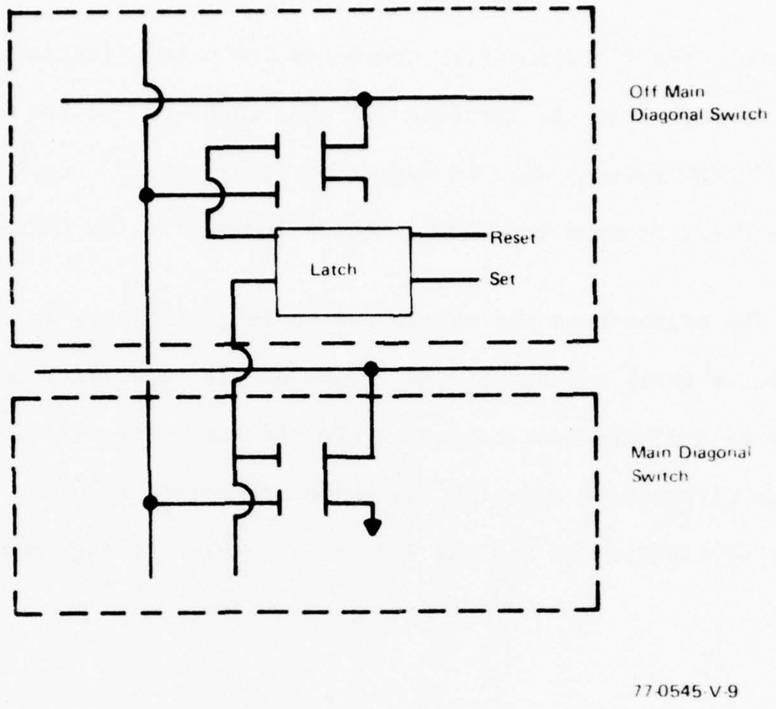


Figure 2-10. Switches within the Target Sorter

latch switch shown in Figure 2-10 is reset such that initially only the main diagonal switches can be enabled. A reset input to each i, j latch switch will be controlled by the outputs from the vector A^1 and B^1 modules shown in Figure 2-8. Each module will only contain one element with the highest subscript like the vector in the CCDM module. Existence of non-zero elements in the vectors A^1 and B^1 modules will only occur when the Coloring Operator detects two adjacent colors during the target painting. That is, when $A_1^1 \neq B_1^1 \neq 0$ a condition which represents multi-color in one target. A flag for such a condition is provided by the CO and will enable the A^1 and B^1 modules for one clock period. The set inputs to the latch switches are gated such that only one latch is enabled for any combination of vectors A and B.

Once the jth Classifier completes the classification, it will reset the jth line such that all latching switches connected to the jth line (excluding the jth CCDM switch) will be open. Moreover, the jth color will be returned to the Color Storage and Memory Module located in the Color Operator Unit.

The approach to the matrix switch is preliminary in nature. In the next quarter we shall concentrate on expanding the capability to handle more than one color equivalent statement within the same object region. Another effort will be directed to reducing the number of colors required which reduce the number of classifiers and the size of the color storage memory module.

2.2 Feature Extraction

A tentative list of features which Maryland will use for classification includes perimeter match, area, perimeter extent, average gray level, and maximum height and width. The purpose of this section is to discuss the implementation of a feature extractor.

2.2.1 Analysis

At this point in the data flow, we may assume that the object has been segmented and the existence of more than one color within the object has been flagged. Also the object in its entirety has been sent to one of the K Classifiers discussed previously. These Classifiers really consist of two parts: a Feature Extractor and an Identifier. Maryland has produced a tentative set of features thus far so the hardware effort will be limited, at present, to those features. We shall continue with the assumption that the image is entering the Classifiers one horizontal line at a time, the processing is moving up the image, and from right to left.

2.2.2 Area

Assume an object shape such as that shown in Figure 2-11.

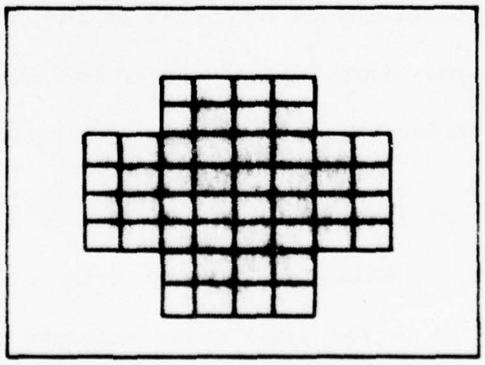


Figure 2-11. Arbitrary Target

The area of the object can be obtained simply by summing the number of pixels within the target; in this case the area is 48 pixels. The Area Feature Extractor then is seen in Figure 2-12.

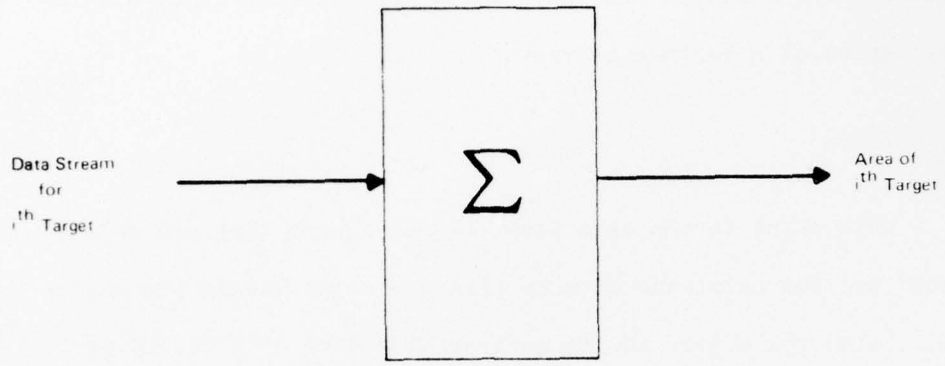


Figure 2-12. Area Feature Extractor 77-0545-V-11

2.2.3 Perimeter Extraction

Both the Perimeter Match and Extent may be considered together. Each pixel has eight neighbors; a perimeter point is defined as having at least one zero neighbor, similarly an interior point is defined as having eight non-zero neighbors. Recall that these definitions apply to the binary image which was developed prior to the Connected Components Algorighm. However, in the Classifier Module no exterior neighbors would be allowed to enter, so the test of a perimeter pixel may be conducted as the complement of an interior target pixel. That is, only target pixels enter the Classifier; these target pixels which are not interior target points must be perimeter target pixels.

The Perimeter Feature Extractor would consist of three lines of memory, each 525 pixels long corresponding to the image frame width. The center pixel would be the location of the pixel being examined for neighbors.

An interesting problem is horizontally aligning these three lines of memory within the Classifier in the same way they are aligned in the image. This can be accomplished by using three lines of memory forming a serpentine and injecting a bit into the delay lines for the appropriate color location. Non destructive readouts form the 3 x 3 moving neighborhood as the pixels shift through as shown in Figure 2-13.

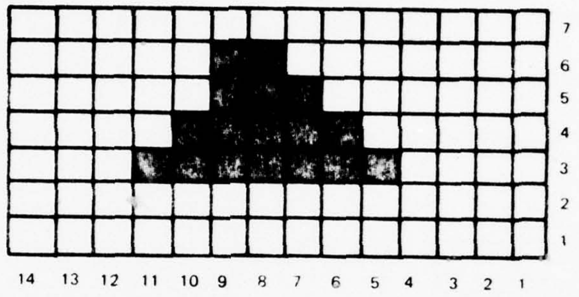


Figure 2-13a. Image

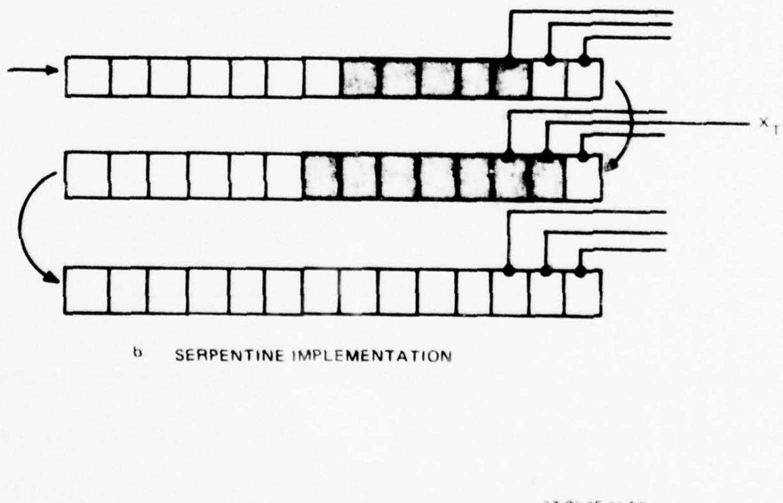


Figure 2-13b. Serpentine Implementation

The second to last pixel in the second row, X_T, is always the pixel position being tested as a perimeter point. This implementation does not have to make special allowances for pixels located on the edges of a target. This is the same technique described in the second and third quarterly reports for forming the moving windows for Gradient Operator, Median Filter, and Non Maximum Suppression. We can AND the neighbors of X_T, complement the output and AND it with X_T. If the output of the first AND is zero, then one of X_T's neighbors is zero and X_T is a perimeter point. See Figure 2-14for the logic. This is fed to a summer and the total number of perimeter points for a target is produced.

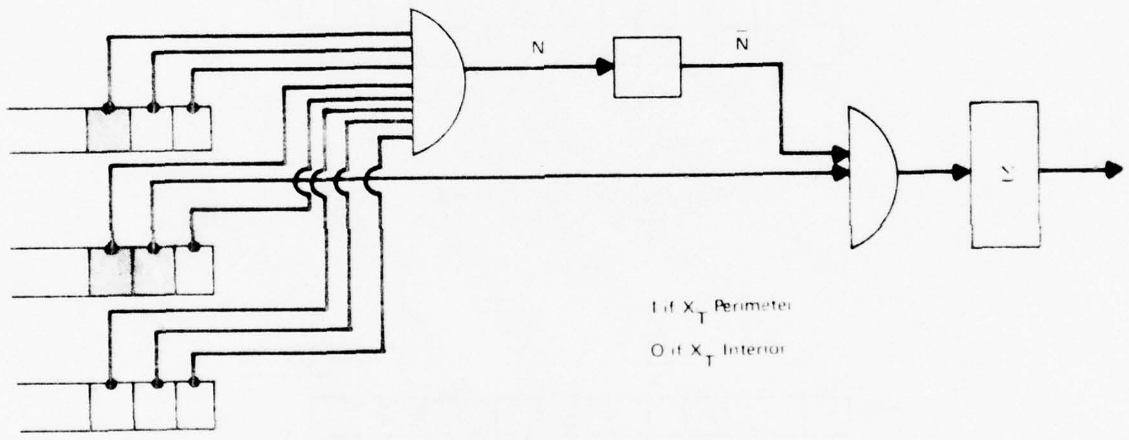


Figure 2-14. Perimeter Point Logic

For the Perimeter Match, the important thing is to clock the image of edges at the same rate that the perimeter extent of the same target is determined. Then X_T is compared with its geometric counterpart in the edge domain, i.e., at the same x, y position in the image. Both are fed to an AND gate and if non-zero, will increment another summer which keeps track of the Perimeter Match Score.

2.2.4 Maximum Target Height and Width

This has to do with keeping track of the maximum target excursions in x and y. For the maximum x extent suppose we use the same three delay line memory which was used for Perimeter Match. We further assume a more complicated object as shown in Figure 2-15. The left most site of second row and the complement of the right most site of the last row are ANDed. If a one comes from the AND gate the summer is incremented by one. Essentially, we count the number of pixels in the first row of the image and then determine the amount of extension in the following rows and add it to the first sum. Note that number indicated by the summer corresponds to a count of the number image columns in the first rows.

The y extent of the image can be determined by counting the number of pixels passing through the three line delay and dividing the number of pixels per line.

2.2.5 Average Gray Level

This involves clocking the Median Filtered image at the same time the color elements are clocked through the Feature Ex-ractor. The gray level sum is obtained and divided by the area obtained earlier by the Feature Extractor. The quotient yields the average gray level.

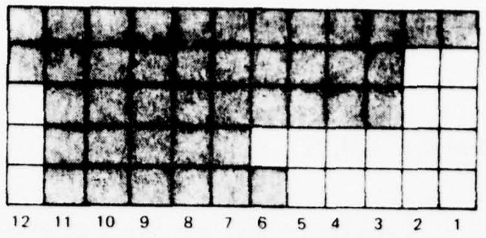


Figure 2-15a. Object

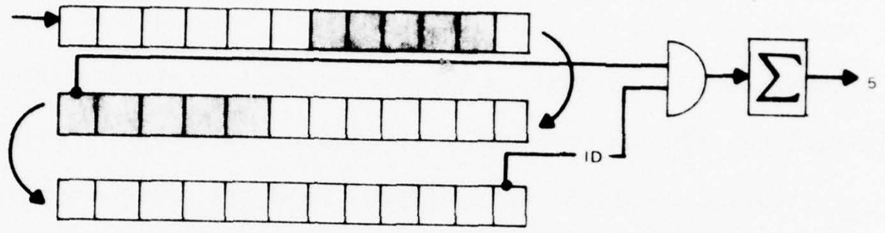


Figure 2-15b. Tapping First Image Row

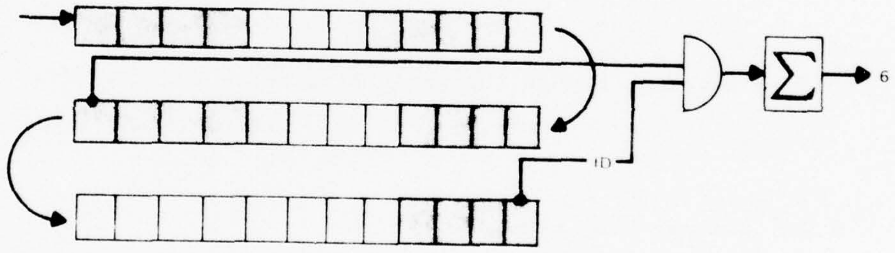
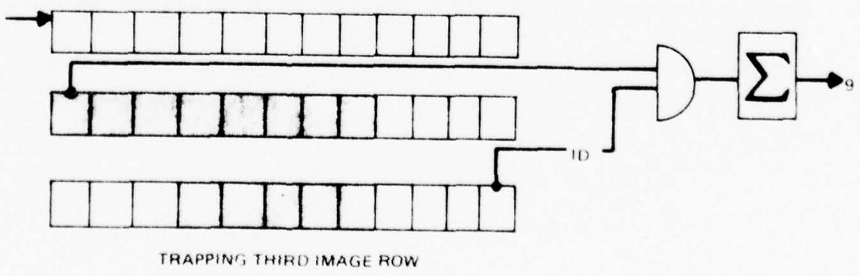


Figure 2-15c. Tapping Second Image Row



77-0545 V 14

Figure 2-15d. Tapping Third Image Row

3.0 HARDWARE FABRICATION

In previous work, we have described the hardware implementation of a number of algorithms. In this section we shall take that work a step further and consider the fabrication of these implementations. Specifically, we shall consider chip size, cryogenic problems, speeds, yields and power consumption relevant to the Non-Maximum Suppression Operator and the Connected Components Algorithm.

3.1 Non-Maximum Suppression Operator

Estimating the physical size of the NMSO will be consistent with the design rules followed for estimating the fabrication of primitive operators reported in previous quarterly reports. It should be restated that the estimates provided are preliminary and adjustments are expected when chip layouts are made. Estimation will be made by first decomposing the operator into its components and calculating the physical dimensions of each port.

The NMSO is made up of several components. A serial input/parallel output CCD structure will be used in quantizing the analog signals injected into the NMSO. The size of this component will be 5 mils x 100 mils wide. The output from the quantizing module will go into a sorter which will be 13 mils by 100 mils in size. The output from the sorter will be injected into a CCD compare register which will compare two signals. One signal is the largest, x, of the neighboring gradient pixels; the other is gradient pixel of interest, y, (see Figure 1-2 in the Jan. 31, 1977 report). The difference output will control the NMSO output according to the following rule

$$\text{NMSO Output} = \begin{cases} 0 & y \leq x \\ y & y > x \end{cases} .$$

Totaling the area required for each NMSO component, we obtain an area of 100 mils by 25 mils. This chip will have MOS structures. We estimate a 25 milliwatt power drain by the NMSO neglecting peripheral clocking and logic. A 10% yield for this chip is predicted.

3.2 Connected Component Algorithm

The Color Sorting Operator is the first intelligent operator we have conceptually developed. In the text describing the workings of the CSO we deliberately did not specify the number of colors required. For a large number of targets per line of image, a corresponding large number of colors will be required. Clearly the organization of the CSO does not depend on the number or size of targets expected in the image plane. However, the size, power consumption, and chip yield do depend on these parameters. To facilitate the estimation of CSO size we shall make certain assumptions. The number of targets handled at any one time shall be less than 10. Each target will extend less than 100 elements in any direction.

The coloring circuit of the CSO consists of several blocks; the coloring logic circuits, the color delay line, and the color bank. We estimate that the size of the coloring logic will require an area of 10 mils x 50 mils. The logic will be MOS structure to maintain compatibility with the other components. A color delay line will be 10 elements wide by one horizontal TV line long. Consistent with the area required for previously described serpentine memory chips, we predict a 400 x 15 mil size. The color bank structure will be very similar to the Median Filter already described. Accounting for the number of quantization levels, we obtain a 25 mil x 25 mil chip. It should be emphasized that the structure of the color bank chip will be facilitated by the development work on the Median Filter.

The target size will determine the length of the delay line between the coloring circuit and the color sorting matrix switch, since the target must be completely painted when the target sorting matrix operates. We have assumed a target size of 100 pixels. The configuration of this delay line will be such as to minimize the number of transfers and thereby limit the MTF. A serial in/parallel out - parallel in/serial out structure will achieve a minimum number of transfers of 98 plus 200, the number of transfers in two horizontal lines. We estimate the size of such a delay line will be 1000 mils by 150 mils for 680 elements long x 100 elements wide.

The switching Matrix in the Target Sorter is estimated to require less than 10 x 10 mil area for each switching mode leading to a total size of 100 x 100 mils. The peripheral logic for controlling the matrix will be two modules 150 x 150 mils in size. For a monolithic Connected Components Algorithm, the estimated size is less than 1000 x 200 mils.

The complexity of the Connected Components Algorithm and its size is not insignificant. Development should proceed in steps such that individual components are fabricated separately. A hybrid configuration can be realized by interconnecting all the component chips. With experimental evaluation and testing, progress toward a monolithic version can be realized. The power and yield for the operation and fabrication of the Connected Components Algorithm is difficult to estimate. An alternative approach is to estimate these parameters for the individual components which are comparable in size and complexity to the primitive operators already discussed. Accordingly, we feel that the power consumption and yield for each component will be about 10-30 milliwatts and 10-20% respectively.

4.0 FOCAL PLANE AREA

This section presents a preliminary estimate of the focal plane area occupied by the portions of the cueing systems developed thus far, i.e., the image has been smoothed (Median Filter), edges obtained (Gradient Operator), edge width reduced (Non-Maximum Suppression), the image has been segmented into targets (Connected Components), and features extracted (area, height, width, perimeter extent, average gray level). The estimate is preliminary in the sense that none of the clocking circuitry has been included in area estimates for the operators. The reason is that methods by which the algorithms will handle edges of the image frame have not been specified. The estimate also does not include the Classifiers.

Assuming that the focal plane is divided into 20 columns. Table 4-1 shows the number of processors required for a system data rate of image pixel/sec. It also shows the geometric area required for each processor and an estimate of the area as defined above. The area thus far is an inch x 3/4 inch.

TABLE 4-1. PRELIMINARY ESTIMATE OF FOCAL PLANE AREA

PROCESS	NUMBER REQUIRED	UNIT AREA	TOTAL AREA
Serpentine Delay (36 pixels long)	400	1000 mils x 1 mil for 20	1000 mils x 20 mils
Median Filter	20	100 mils x 128 mils for 1	400 mils x 640 mils
Gradient Operator	20	8 mils x 10 mils for 1	32 mils x 50 mils
Non-Maximum Suppression	20	100 mils x 25 mils for 1	300 mils x 200 mils
Connected Components	1	1000 mils x 200 mils for 1	1000 mils x 200 mils
Line Delay			
			1 inch x 3/4 inch.

APPENDIX A SIMPLIFIED DESCRIPTION OF CCD CIRCUITS FOR AUTOMATIC TARGET CUEING

A1.0 INTRODUCTION

The purpose of this paper is to describe the CCD circuits that Westinghouse has used in implementing the Maryland cueing algorithms. The implementation of three algorithms is described; Gradient Operator, Median Filter and Non-Maximum Suppression.

A2.0 GRADIENT OPERATOR

The Gradient Operator is an edge detector which is defined as GRAD OP = max $\{|A-B|, |C-D|\}$ where A, B, C and D each represent the sums of overlapping regions of 4 x 4 pixels each, as seen in Figure A2-1.

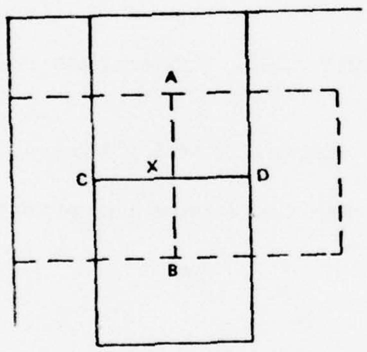


Figure A2-1. Gradient Operator

The value of GRAD OP is placed in the pixel location marked "x" which is one pixel to the left and above the center of the entire region. The key arithmetic operation in GRAD OP is the formation of the difference

$$A-B = \begin{cases} 0 & \text{if } |A| < |B| \\ A-B & |A| > |B| \end{cases} \qquad \text{(A-1)}$$

which is realized on a silicon substrate with the configuration shown in

Figure A2-2. D_{in} is a diffusion diode through which charge is injected into the chip; A and B are gates whose potentials are controlled by voltages representing the sums A and B respectively. These gates will form a trap to retain some of the injected charge. The trapped charge is equal to A-B and is removed by the transfer gate.

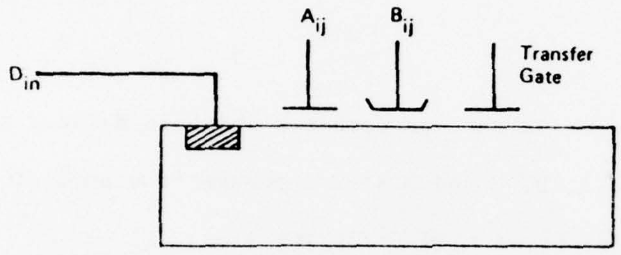


Figure A2-2. Subtraction Module

Let us consider a physical analogy in which Din, A, B, and TG are a set of steps, the height of each step represents the potential, and the charge is represented as water.

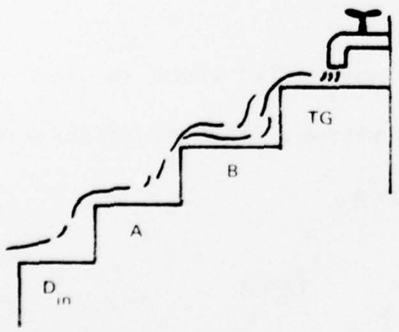


Figure A2-3a. No Water Trapped, |B|>|A|

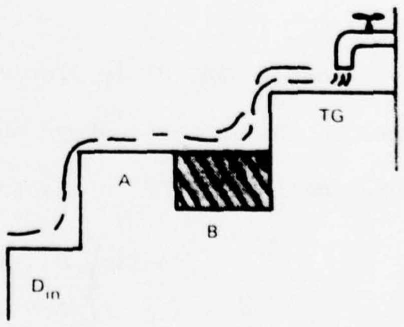


Figure A2-3b. Watter Trapped |B|<|A|

when the height (potential) of step (gate) A is raised above that of step (gate) B, water (charge) is trapped above B in an amount corresponding to A-B. Figure A 2-3 a, b represents both conditions of Eqn. (A-1).

The algorithm $$B-A = \begin{cases} 0 \text{ if } |B| < |A| \\ B-A \text{ if } |B| > |A| \end{cases}$$

requires another silicon substrate in which the gate positions of A and B are reversed. The block diagram of the absolute difference operator $|A-B|$ is shown in Figure A2-4.

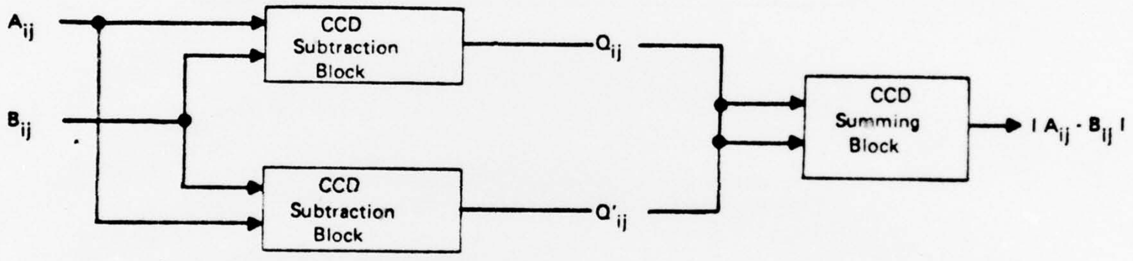


Figure A2-4. Absolute Value Circuit

If $|A| < |B|$, then $B_{out} = 0$, $A_{out} = B-A$ and $A_{out} + B_{out} = B-A$. Similarly, if $|A| > |B|$, then $B_{out} = A-B$, $A_{out} = 0$ and $A_{out} + B_{out} = A-B$. These two statements form an absolute value operator. A similar operation can be formed for $|C-D|$, but GRAD OP really compares $|A-B|$ and $|C-D|$ and chooses the maximum since it is defined as max $\{|A-B|, |C-D|\}$. Figure A2-5 shows the GRAD OP block diagram and Figure A2-6 shows the outputs at various locations.

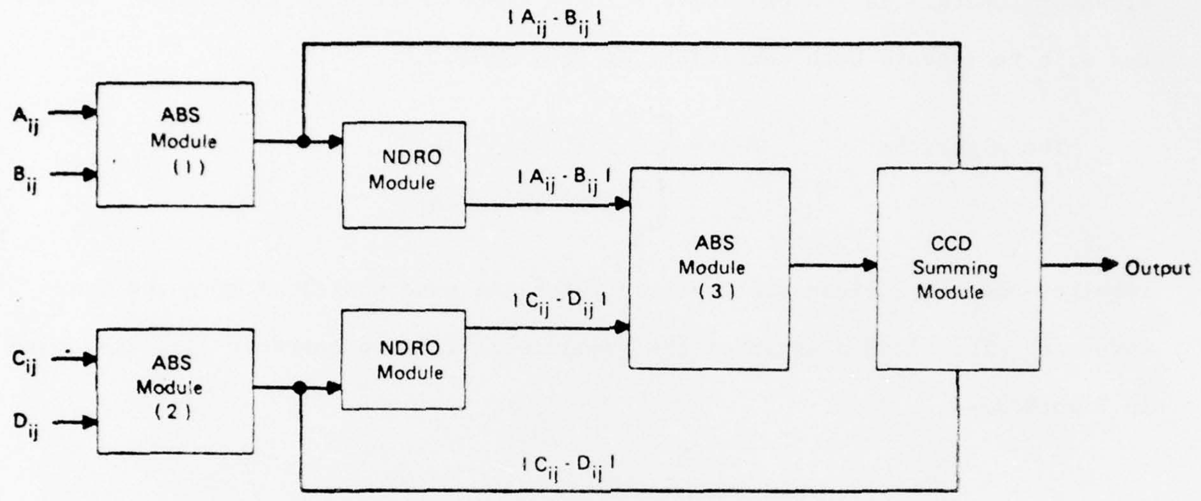


Figure A2-5. CCD Gradient Operator

Condition	$\|A-B\| < \|C-D\|$	$\|C-D\| < \|A-B\|$
ABS Module (3) Output	$\|C-D\| - \|A-B\|$	$\|A-B\| - \|C-D\|$
Summing CCD input	$\|C-D\| + \|A-B\|$	$\|C-D\| - \|A-B\|$
Outputs From Summing CCD	$2\|C-D\|$	$2\|A-B\|$

Figure A2-6. Output of GRAD OP at Various Locations

A3.0 MEDIAN FILTER

The Median Filter acts to extract the median value from a 5 x 5 pixel window and place that value in the center pixel location; the Filter can be considered as a smoothing operator. The Filter quantizes each of the 25 analog signals into a number of discrete units and then sorts the 25 quantized signals by arranging them in a descending order of magnitude.

A3.1 Charge Quantizer

The silicon substrate forming the Quantizer is shown in Figure A-31.

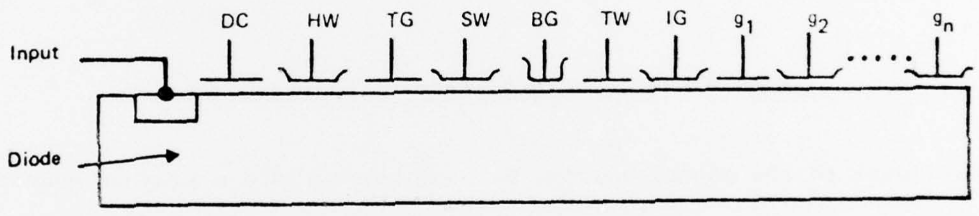
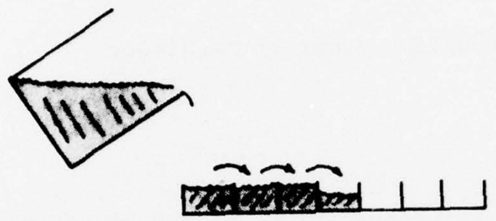


Figure A3-1. Charge Quantizer. Symbols are Identified in the Text

D_{in} is the diffusion diode through which charge is injected into the chip and into the holding well, HW. DC blocks the charge from leaving via D_{in}. An amount of charge, Q, proportional to S, the signal voltage, is removed from HW via the transfer gate, TG, and placed in the signal well, SW. Via the blocking gate, BG, and the thimble well, TW, a discrete amount of charge, q, is removed and placed in well g_1. Another quantum q is removed from SW, and placed in g_1 while the first charge has been shifted to g_2. This process is repeated until SW is empty and all the charge has been placed in a number of discrete wells g_1, g_2 .... g_n.

Again referring to a physical analogy, Figure A3-2 shows a container with an amount of water (charge),



77-0545-V-16

Figure A3-2. Flow Analogy to Charge Quantizer

proportional to the signal voltage S,being poured into a tray of quantized bins. When a bin is filled with water the water flows over the top into the next bin. The volume of water is divided between a number of discrete bins.

A3.2 Sorter

Recall that the contents of wells g_1, g_2, ... g_n of the quantizer (Figure A3-1) each contain, at most, an amount of charge q. The contents of each well is parallel shifted into its corresponding channel of the sorter (Figure A3-3), so that if there were q charge in g_3, there is now q charge in I_3 and so on. Forming traps as described in Section A2.0 with wells P_{g1}, P_{g2}, and P_{g3},the charge in each channel is shifted into the large holding well (LHW). The large holding well is partitioned into N channels also. Consider a numerical example; a sequence of numbers 4, 7, 5 is quantized at q = 1 so that 4, 7, and 5 levels respectively represent each number. Then Figure A3-4 shows the sequence as it goes through the quantizer; the g_1, g_2, ... g_n wells; the I_1, I_2, ... I_n

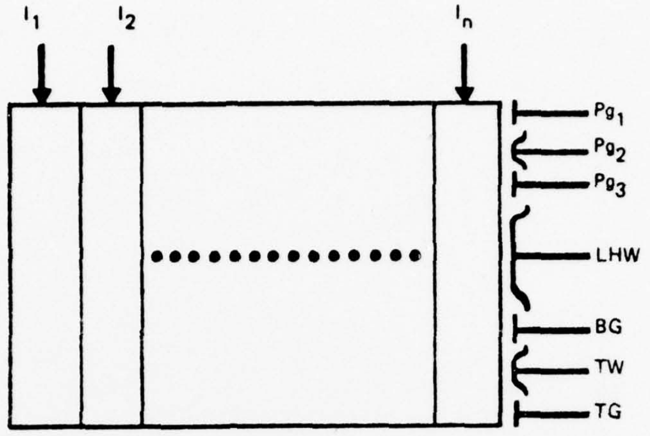


Figure A3-3. CCD Sorter

channels; and the large holding well. It also shows the removal sequence from the large holding well and the remainder at each stage. The numbers are removed in order of decreasing magnitude 7, 5, 4 which shows the numbers have been sorted by magnitude.

A4.0 NON-MAXIMUM SUPPRESSION

The Gradient Operator extracts edges in either the horizontal or vertical direction; the Non-Maximum Suppression Algorithm then looks in a direction perpendicular to the edge for a larger gradient. If a larger value cannot be found, the edge under consideration is retained; the edge is removed if a larger value is found. The Algorithm is shown in Figure 4-1.

	1	2	3	4	5	6	7	8	9 .... n
g_i Wells									
5	q	q	q	q	q				
7	q	q	q	q	q	q	q		
4	q	q	q	q					
I_i Channels									
5	q	q	q	q	q				
7	q	q	q	q	q	q	q		
4	q	q	q	q					
Large Holding Well									
4	q	q	q	q					
4,7	2q	2q	2q	2q	q	q	q		
4,7,5	3q	3q	3q	3q	2q	q	q		
First Removal									
	q	q	q	q	q	q	q		
Remainder									
	2q	2q	2q	2q	q				
Second Removal									
	q	q	q	q	q				
Remainder									
	q	q	q	q					
Third Removal									
	q	q	q	q					
Remainder									

Figure A3-4. Sorting Sequence

```
x x x

x x x

  x                    If any x>y, y = 0

  y      →             Otherwise retain y

  x                    x, y are gradient values

x x x

x x x
```

Figure A4-1. Arrangement of Comparison for Non-Maximum Suppression

The gradient under consideration is a vertical one and the area examined for larger gradients is in the vertical direction.

Embodiment of the Non-Maximum Suppression Algorithm (NMS) requires severa operations with CCD structures. A key part of NMS is extracting the largest x_m gradient value in the neighborhood surrounding y; x_m is then compared to the gradient value y_g representing the y^{th} pixel. Sorting the x values to obtain x_m can be accomplished by the sorting operation described earlier. Comparing x_m and y_g can be done by the subtraction module also described before. Then the block diagram appears in Figure A4-2.

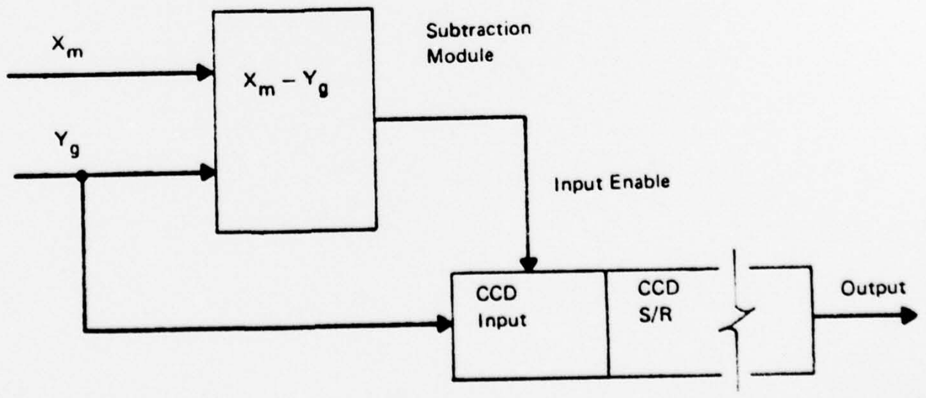


Figure A4-2. NMS Block Diagram

This time the subtraction module outputs an enable signal to the CCD shift register instead of the actual difference. A blocking gate shown in Figure A4-3 is used to block (enable) y_g from entering the register.

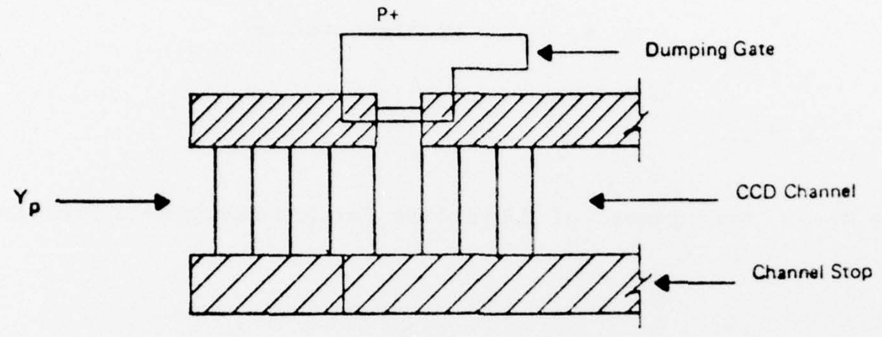


Figure A4-3. Blocking Gate